Topic 13
procedural design and Strings

“Ugly programs are like ugly suspension bridges: they’re much more liable to collapse than pretty ones, because the way humans (especially engineers) perceive beauty is intimately related to our ability to process and understand complexity.”
- Eric S. Raymond,
Author of The Cathedral and the Bazaar

Copyright Pearson Education, 2010
Based on slides by Marty Stepp and Stuart Reges
from http://www.buildingjavaprograms.com/

Nested if/else question

Formula for body mass index (BMI):

\[
BMI = \frac{weight}{height^2} \times 703
\]

Write a program that produces output like the following:

This program reads data for two people and computes their body mass index (BMI) and weight status.

Enter next person's information:
height (in inches)? 73.5
weight (in pounds)? 230
BMI = 29.93
overweight

Enter next person's information:
height (in inches)? 71
weight (in pounds)? 220.5
BMI = 30.75
obese
Difference = 0.82

One-person, no methods

import java.util.*;
public class BMI {
    public static void main(String[] args) {
        System.out.println("This program reads ... (etc.)");
        Scanner console = new Scanner(System.in);
        System.out.println("Enter next person's information:");
        System.out.print("height (in inches)? ");
        double height = console.nextDouble();
        System.out.print("weight (in pounds)? ");
        double weight = console.nextDouble();
        double bmi = weight * 703 / height / height;
        System.out.printf("BMI = %.2f
", bmi);
        if (bmi < 18.5) {
            System.out.println("underweight");
        } else if (bmi < 25) {
            System.out.println("normal");
        } else if (bmi < 30) {
            System.out.println("overweight");
        } else {
            System.out.println("obese");
        }
    }
}

"Chaining"

- main should be a concise summary of your program.
  - It is bad if each method calls the next without ever returning (we call this chaining):

- A better structure has main make most of the calls.
  - Methods must return values to main to be passed on later.
### Bad "chain" code

```java
public class BMI {
    public static void main(String[] args) {
        System.out.println("This program reads ... (etc.)");
        Scanner console = new Scanner(System.in);
        person(console);
    }

    public static void person(Scanner console) {
        System.out.println("Enter next person's information:");
        System.out.print("height (in inches)? ");
        double height = console.nextDouble();
        getWeight(console, height);
    }

    public static void getWeight(Scanner console, double height) {
        System.out.print("weight (in pounds)? ");
        double weight = console.nextDouble();
        computeBMI(console, height, weight);
    }

    public static void computeBMI(Scanner s, double h, double w) {
        // ...
    }
}
```

### Procedural heuristics

1. Each method should have a clear responsibility.
2. No method should do too large a share of the overall task.
3. Minimize coupling and dependencies between methods.
4. The main method should read as a concise summary of the overall set of tasks performed by the program.
5. Variables should be declared/used at the lowest level possible.

### Better solution

```java
// This program computes two people's body mass index (BMI) and compares them.
// The code uses Scanner for input, and parameters/returns.
import java.util.Scanner;

public class BMI {
    public static void main(String[] args) {
        introduction();
        Scanner console = new Scanner(System.in);
        double bmi1 = person(console);
        double bmi2 = person(console);
        // report overall results
        report(1, bmi1);
        report(2, bmi2);
        System.out.println("Difference   = "+Math.abs(bmi1 - bmi2));
    }

    // prints a welcome message explaining the program
    public static void introduction() {
        System.out.println("This program reads ...");
    }
}
```

### Better solution, cont'd.

```java
// reads information for one person, computes their BMI, and returns it
public static double person(Scanner console) {
    System.out.println("Enter next person's information:");
    System.out.print("height (in inches)? ");
    double height = console.nextDouble();
    System.out.print("weight (in pounds)? ");
    double weight = console.nextDouble();
    return bmi(height, weight);
}

// Computes/returns a person's BMI based on their height and weight.
public static double bmi(double height, double weight) {
    return weight * 703 / (height * height);
}

// Outputs information about a person's BMI and weight status.
public static void report(int number, double bmi) {
    System.out.printf("Subject%5dBMI = %.2f\n", number, bmi);
    if (bmi < 18.5) {
        System.out.println("underweight");
    } else if (bmi < 25) {
        System.out.println("normal");
    } else if (bmi < 30) {
        System.out.println("overweight");
    } else {
        System.out.println("obese");
    }
}
```
Strings

- **String**: An object storing a sequence of text characters.
  - Unlike most other objects, a String is not always created with `new`.

```java
String name = "text";
String name = expression;
```

- Examples:
  ```java
  String name = "Marla Singer";
  int x = 3;
  int y = 5;
  String point = "(" + x + ", " + y + ")";
  ```

Indexes

- Characters of a string are numbered with 0-based *indexes*:

```java
index 0 1 2 3 4 5 6 7
character K . S c o t t t
```

- First character's index : 0 (zero based indexing)
- Last character's index : 1 less than the string's length
- The individual characters are values of type `char` (another primitive data type)

String methods

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexOf(str)</td>
<td>index where the start of the given string appears in this string (-1 if not found)</td>
</tr>
<tr>
<td>length()</td>
<td>number of characters in this string</td>
</tr>
<tr>
<td>substring(index1, index2)</td>
<td>the characters in this string from index1 (inclusive) to index2 (exclusive); if index2 is omitted, grabs till end of string</td>
</tr>
<tr>
<td>toLowerCase()</td>
<td>a new string with all lowercase letters</td>
</tr>
<tr>
<td>toUpperCase()</td>
<td>a new string with all uppercase letters</td>
</tr>
</tbody>
</table>

- These methods are called using the dot notation:

```java
String student = "Olivia Scott";
System.out.println(student.length());  // 12
```

String method examples

```java
// index 012345678901
String s1 = "Olivia Scott";
String s2 = "Isabelle Scott";
System.out.println(s2.length());  // 14
System.out.println(s1.indexOf("e"));  // -1
System.out.println(s2.indexOf("e"));  // 4
System.out.println(s1.substring(7, 10)); // "Sco"
String s3 = s2.substring(4, 10);
System.out.println(s3.toLowerCase()); // "elle s"
```

- Given the following string:

```java
// index 0123456789012345678901
String book = "Building Java Programs";
```

- How would you extract the word "Building"?
  - (Write code that can extract the first word from any string.)
Clicker 1

- What is output by the following code?

```java
String s1 = "Football";
String s2 = s1.substring(4, 8);
s2.substring(1);
System.out.print(s2);
```

A. Football  
B. ball  
C. all  
D. No output due to syntax error.  
E. No output due to runtime error.

---

Clicker 2

- What is output by the following code?

```java
String s1 = "taxicab";
String s2 = "acables";
String s3 = s1.substring(4);
String s4 = s2.substring(1, 4);
if (s3.length() == s4.length())
    System.out.print("1");
else
    System.out.print("2");
if (s3 == s4)
    System.out.print("1");
else
    System.out.print("2");
```

A. 11  
B. 12  
C. 21  
D. 22  
E. No output due to syntax error.
Comparing Strings

- Relational operators such as < and <= are undefined on objects in Java.
- == is defined but normally doesn’t work as intended

```java
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name == "Barney") {
    System.out.println("I love you, you love me,");
    System.out.println("We're a happy family!");
}
```

- This code will compile, but it will not print the song.
- == compares objects by references (seen later), so it often gives false even when two Strings have the same letters.

String test methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>equals(str)</code></td>
<td>whether two strings contain the same characters</td>
</tr>
<tr>
<td><code>equalsIgnoreCase(str)</code></td>
<td>whether two strings contain the same characters,</td>
</tr>
<tr>
<td></td>
<td>ignoring upper vs. lower case</td>
</tr>
<tr>
<td><code>startsWith(str)</code></td>
<td>whether one contains other’s characters at start</td>
</tr>
<tr>
<td><code>endsWith(str)</code></td>
<td>whether one contains other’s characters at end</td>
</tr>
<tr>
<td><code>contains(str)</code></td>
<td>whether the given string is found within this one</td>
</tr>
</tbody>
</table>

```java
String name = console.next();
if (name.startsWith("Prof")) {
    System.out.println("When are your office hours?");
} else if (name.endsWith("OBE")) {
    System.out.println("Yes Sir!");
}
```

The equals method

- Objects are compared using a method named `equals`.

```java
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name.equals("Barney")) {
    System.out.println("Fred's Friend.");
    System.out.println("Purple Dinasaur.");
    System.out.println("In trouble.");
}
```

- The equals method returns a value of type boolean, the type used in logical tests.

Strings questions

- Write a method to determine if a String is a possible representation of a DNA strand
  - contains only A, C, T, and G
- Write a method to create a Watson-Crick complement given a String that represents a strand of DNA
  - replace A with T, C with G, and vice versa
- Given a String that represents a strand of DNA return the first substring that exists between "ATG" and either "TAG" or "TGA"
  - no overlap allowed
String Questions

- Write a method that returns the number of times a given character occurs in a String
- Write a method that returns the number of times the punctuation marks . ? ! , : " ; ' occur in a String